Manual

# Feature Introduction

This feature maintains each data version generated during MySQL running, and provides SQL to easily trace back to the historical version.

# SQL

## Create Temporal Table

```
CREATE TABLE [IF NOT EXISTS] tbl_name (create_definition,...)[table_options][WITH TEMPORAL]
    table_options:
        table_option[[,] table_option] ...
    table_option:
        AUTO_INCREMENT [=] value
      | ...
      | HIST_TABLESPACE [=] value
```

- **WITH TEMPORAL**: Create two tables at the same time, *tbl_name* stores the latest data versions and is called ***current table***. *tbl_name_history* stores the historical versions and is called ***history table***.
- **HIST_TABLESPACE** [=] value：The tablespace of *tbl_name_history*. The default ***ibhistory*** tablespace is used if HIST_TABLESPACE is not given.
- CREATE LIKE, CREATE SELECT, TEMPORARY and PARTITION are not supported.

- name of history table = name of current table + *_history* postfix
- The schema of history table is the same as the one of current table.
- DML/DDL/DQL on history table and DDL on current table are prohibited.
- History and current tables sharing a single tablespace is not recommended.
- DML on current table results in old versions to be restored in history table.

Example：

```
mysql> CREATE TABLE t(a INT PRIMARY KEY, b INT) HIST_TABLESPACE=ibhistory WITH TEMPORAL;
Query OK, 0 rows affected (0.02 sec)

mysql> SHOW TABLES;
+-----------------+
| Tables_in_testdb |
+-----------------+
| t               |
| t_history       |
+-----------------+
2 rows in set (0.00 sec)
mysql> DESC t;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | NO   | PRI | NULL    |       |
| b     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> DESC t_history;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | NO   | PRI | NULL    |       |
| b     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Create table

```
mysql> SELECT * FROM t_history;
ERROR 4569 (HY000): Can not access history table "(t_history)" directly.
mysql> UPDATE t_history SET b=b+1;
ERROR 4569 (HY000): Can not access history table "(t_history)" directly.
mysql> DROP TABLE t_history;
Query OK, 0 rows affected (0.01 sec)
```

History table can’t be accessed directly

```
mysql> INSERT INTO t VALUES(1,1), (2,2), (3,3);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> UPDATE t SET b=b+1 WHERE a=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELETE FROM t WHERE a=2;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t;
+---+------+
| a | b    |
+---+------+
| 1 |    2 |
| 3 |    3 |
+---+------+
2 rows in set (0.00 sec)
```

DML on current table

## Temporal Query

Temporal query retrieves data versions from current and history table. see SQL:2011 for more about temporal query.

```
SELECT
    [ALL | DISTINCT | DISTINCTROW] ...
    select_expr [, select_expr ...]
    [FROM table_references [WHERE where_condition] ...]

    table_references:
        tbl_name [[AS] alias] [index_hint_list] [temporal_hint]

    temporal_hint:
        SYSTEM TIME AS OF value [ONLY HISTORY]
      | SYSTEM TIME FROM value1 TO value2 [ONLY HISTORY]
      | SYSTEM TRANSACTION value [ONLY HISTORY]
```

- *tbl_name* is a current table.
- *temporal_hint* is only supported in *select*. *table_references* with *temporal_hint* in *update* or *delete* raises error.
- *ONLY HISTORY* queries history table only, while current table is skipped.
- Illustrate the semantics of temporal_hint with the diagram below.
  A sketch diagram is given for the purpose of illustration and is not what MySQL client displays.

| Pk | Version | Begin Tid | End Tid | Begin ts | End ts |
|---|---|---|---|---|---|
| 1 | V0 | 2000 | 2003 | T0 | T1 |
| 1 | V1 | 2003 | 2004 | T1 | T3 |
| 1 | V2 | 2004 | | T3 | |

*Pk* represents a user record.
*Version* represents a unique data version of the user record.
*Begin Tid* and *End Tid* are the transactions which created and removed the data version.
*Begin ts* and *Edn ts* are the timestamp when *Begin Tid* and *End Tid* were committed.

- AS OF value. A time-point query retrieves the data version which was/is in current state at the *value* time. For example,
  *select * from t system time as of T2 where pk=1* gets V1
  *select * from t system time as of T1 where pk=1* gets V1. V0 is a history version at T1.
- FROM value1 TO value2. A time-range query retrieves the data versions was/is in current state during value1~value2 time. It requires a data version's lifetime overlaps the given time range.
  *select * from t system time from T0 to T2 where pk=1* gets V0 and V1, because [V0.BeginTs, V0.EndTs) ∩ [T0, T2] ≠∅, [V1.BeginTs, V1.EndTs) ∩ [T0, T2] ≠∅
- TRANSACTION value. Transaction ID query retrieves the data versions created or removed by this transaction.
  *select * from t system transaction 2003 where pk=1* gets V0 and V1. Transaction 2003 removed V0 and created V1.

- MySQL Client Examples

Time-Point Query

Data versions in the current table are (1,2), (3,3)

Data versions are (1,2), (3,3), (2,2) at 11:57:48, *update t set b=b+1 where a=1* has been committed then, while *delete from t where a=2* has not been done.

11:57:40 时刻版本为(3,3),(1,1),(2,2), 当时 update 与 delete 都未执行

Data versions are (3,3), (1,1), (2,2) at 11:57:40, update and delete have been done then.

```
mysql> SELECT * FROM t;
+---+------+
| a | b    |
+---+------+
| 1 |    2 |
| 3 |    3 |
+---+------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM t SYSTEM TIME AS OF '2020-02-03 11:57:48';
+---+------+
| a | b    |
+---+------+
| 1 |    2 |
| 3 |    3 |
| 2 |    2 |
+---+------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM t SYSTEM TIME AS OF '2020-02-03 11:57:40';
+---+------+
| a | b    |
+---+------+
| 3 |    3 |
| 1 |    1 |
| 2 |    2 |
+---+------+
3 rows in set (0.00 sec)
```

Time-Range Query:

a. All date versions are (1,2), (3,3), (1,1), (2,2) from 11:00:00 to 12:00:00.
b. With ONLY HISTORY keywords, (1,1), (2,2) are retrieved from history table. (1,2) and (3,3) are not returned because they are in current table.

```
mysql> SELECT * FROM  t SYSTEM TIME FROM '2020-02-03 11:00:00' TO '2020-02-03 12:00:00';
+---+------+
| a | b    |
+---+------+
| 1 |    2 |
| 3 |    3 |
| 1 |    1 |
| 2 |    2 |
+---+------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM  t SYSTEM TIME FROM '2020-02-03 11:00:00' TO '2020-02-03 12:00:00' ONLY HISTORY;
+---+------+
| a | b    |
+---+------+
| 1 |    1 |
| 2 |    2 |
+---+------+
2 rows in set (0.01 sec)
```

Transaction ID Query:

Transaction 2675 carried out *update t set b=b+1 where a=1*, which removed (1,1) and created (1,2).

Transaction 2678 carried out *delete from t where a=2*, which removed (2,2).

```
mysql> SELECT * FROM t SYSTEM TRANSACTION 2675;
+---+------+
| a | b    |
+---+------+
| 1 |    2 |
| 1 |    1 |
+---+------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM t SYSTEM TRANSACTION 2678;
+---+------+
| a | b    |
+---+------+
| 2 |    2 |
+---+------+
1 row in set (0.00 sec)
```

Wrong Use:

a. A ordinary table doesn't support temporal query.
b. Non-select statement on a current table doesn't support temporal query.
c. Can not access history table directly.

```
mysql> SELECT * FROM t1 SYSTEM TRANSACTION 2678;
ERROR 4568 (HY000): Only original table supports temporal features.
mysql> UPDATE t SYSTEM TRANSACTION 2678 SET b=b+1 WHERE a=3;
ERROR 4567 (HY000): Only select statement supports temporal features.
mysql> SELECT * FROM t_history SYSTEM TRANSACTION 2678;
ERROR 4569 (HY000): Can not access history table "(t_history)" directly.
```

## Transaction Status Query

Statuses of all transactions are maintained and readable.

| TRXTOTIME(val1, val2, val3) |
|---|

- val1 is transaction id in string format. val2 and val3 are decimals.
- Get the statuses of transactions whose ids are from [val1-val2, val1+val3], i.e., TRXTOTIME("2003", 1, 1) gets transaction 2002, 2003, 2004, and 2005.
- Transaction Status includes 4 fields: TRX_ID is the transaction id. START_TIME is when the transaction began. FINISH_TIME is when the transaction committed or aborted. STATUS is the current status of the transaction, which is one of {INPROGRESS, COMMITTED, ABORTED, UNDO}. INPROGRESS, COMMITTED, ABORTED are as their literal meanings. UNDO means a user transaction that has not been started or an InnoDB internal transaction, which occupies an id but is not maintained by our system.

TIMETOTRX(val1, val2, val3) :

- val1 is of type datetime. val2 and val3 are decimals in seconds.
- Get the statuses of transactions which are committed during [val1-val2, val1+val3], i.e., TIMETOTRX("2020-01-01 00:00:00", 60, 60) gets transactions which are committed from 2019-12-31 23:59:00 to 2020-01-01 00:01:00.

An accurate transaction id is required by temporal transaction ID query, and it can be speculated with TIMETOTRX.

CURRENTTRX()

- Get the max transaction ID for now.

For example,

a. An ongoing transaction 2677 is in INPROCESS state, and its FINISH_TIME is 1970-01-01 08:00:00, which is an unset datetime value.
b. Transaction 2679 is in UNDO state, and its STRAT_TIME and FINISH_TIME are both 1970-01-01 08:00:00, while the max transaction id is 2689, so 2679 is an InnoDB internal transaction.
c. The max transaction id is the next one to be allocated. 2689 and its successors are all in UNDO state.

```
mysql> TRXTOTIME("2678", 1, 1);
+--------+---------------------+---------------------+------------+
| TRX_ID | START_TIME          | FINISH_TIME         | STATUS     |
+--------+---------------------+---------------------+------------+
|   2677 | 2020-02-03 11:57:44 | 1970-01-01 08:00:00 | INPROGRESS |
|   2678 | 2020-02-03 11:57:49 | 2020-02-03 11:57:49 | COMMITED   |
|   2679 | 1970-01-01 08:00:00 | 1970-01-01 08:00:00 | UNDO       |
+--------+---------------------+---------------------+------------+
3 rows in set (0.00 sec)

mysql> TIMETOTRX("2020-02-03 11:57:40", 0, 10);
+--------+---------------------+---------------------+----------+
| TRX_ID | START_TIME          | FINISH_TIME         | STATUS   |
+--------+---------------------+---------------------+----------+
|   2674 | 2020-02-03 11:57:44 | 2020-02-03 11:57:44 | COMMITED |
|   2675 | 2020-02-03 11:57:44 | 2020-02-03 11:57:44 | COMMITED |
|   2678 | 2020-02-03 11:57:49 | 2020-02-03 11:57:49 | COMMITED |
+--------+---------------------+---------------------+----------+
3 rows in set (0.00 sec)

mysql> CURRENTTRX();
+----------------+
| CURRENT_TRX_ID |
+----------------+
|           2689 |
+----------------+
1 row in set (0.00 sec)

mysql> TRXTOTIME("2689", 0, 1);
+--------+---------------------+---------------------+--------+
| TRX_ID | START_TIME          | FINISH_TIME         | STATUS |
+--------+---------------------+---------------------+--------+
|   2689 | 1970-01-01 08:00:00 | 1970-01-01 08:00:00 | UNDO   |
|   2690 | 1970-01-01 08:00:00 | 1970-01-01 08:00:00 | UNDO   |
+--------+---------------------+---------------------+--------+
2 rows in set (0.00 sec)
```

# Get Metadata

No newly introduced syntax.

Current and history flags are added into SHOW CREATE TABLE output and INFORMATION_SCHEMA.TABLES.

For example:
In SHOW CREATE TABLE output, current table is marked as /*FLASHBACK ORIGINAL */, history table is marked as /*FLASHBACK HISTORICAL*/.

```
mysql> SHOW CREATE TABLE t \G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE `t` (
  `a` int(11) NOT NULL,
  `b` int(11) DEFAULT NULL,
  PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 FLASHBACK /*FLASHBACK ORIGINAL*/
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t_history \G
*************************** 1. row ***************************
       Table: t_history
Create Table: CREATE TABLE `t_history` (
  `a` int(11) NOT NULL,
  `b` int(11) DEFAULT NULL,
  PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 /*FLASHBACK HISTORICAL*/
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t1 \G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

In INFORMATION_SCHEMA.TABLES，current table has ORIG_TABLE=YES，history table has HIST_TABLE=YES.

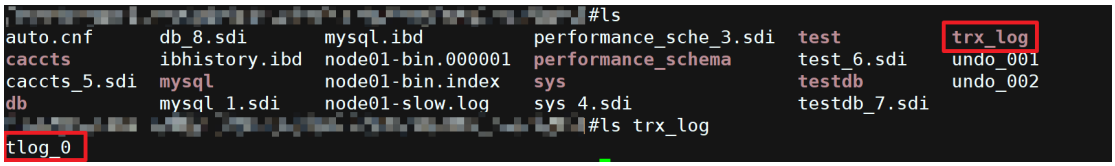
```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, HIST_TABLE, ORIG_TABLE
    ->     FROM INFORMATION_SCHEMA.TABLES
    ->     WHERE TABLE_NAME="t" AND TABLE_SCHEMA="db" \G
*************************** 1. row ***************************
TABLE_SCHEMA: db
  TABLE_NAME: t
  HIST_TABLE: NO
  ORIG_TABLE: YES
1 row in set (0.00 sec)

mysql> SELECT TABLE_SCHEMA, TABLE_NAME, HIST_TABLE, ORIG_TABLE
    ->     FROM INFORMATION_SCHEMA.TABLES
    ->     WHERE TABLE_NAME="t_history" AND TABLE_SCHEMA="db" \G
*************************** 1. row ***************************
TABLE_SCHEMA: db
  TABLE_NAME: t_history
  HIST_TABLE: YES
  ORIG_TABLE: NO
1 row in set (0.00 sec)

mysql> SELECT TABLE_SCHEMA, TABLE_NAME, HIST_TABLE, ORIG_TABLE
    ->     FROM INFORMATION_SCHEMA.TABLES
    ->     WHERE TABLE_NAME="t1" AND TABLE_SCHEMA="db" \G
*************************** 1. row ***************************
TABLE_SCHEMA: db
  TABLE_NAME: t1
  HIST_TABLE: NO
  ORIG_TABLE: NO
1 row in set (0.00 sec)
```

# Transaction Log

Transaction statuses are kept in transaction logs, which locate in datadir/trx_log. Transaction log files are named as tlog_0, tlog_1, tlog_2, ⋯, and each of them is up to 1M.
For example：




# Default History Tablespace

Default history tablespace is used when creating a history table without giving history tablespace.

The default history tablespace is named as ibhistory, and the data file is ibhistory.ibd.

For example：